计 算 机 科 学 丛 书

Data Structures, Algorithms, and Applications in C++

数据结构 算法与应用

—— C++语言描述

（英文版）

（美） Sartaj Sahni 著
佛 罗 里 达 大 学

机械工业出版社
China Machine Press
Mc Graw Hill Education

计算机科学丛书

数据结构算法与应用C++语言描述

（英文版）

Data Structures, Algorithms, and Applications in C++

（美）Sartaj Sahmi 著
(佛罗里达大学)

机械工业出版社
China Machine Press

WCB
McGraw-Hill

本书版权登记号：图字：01-99-0113

图书在版编目（CIP）数据

数据结构算法与应用——C++语言描述：英文/（美）塞尼（Sahni, S）著；-影印版. – 北京：机械工业出版社，1999.3
（计算机科学丛书）
ISBN 7-111-07017-8

Ⅰ.数… Ⅱ.塞… Ⅲ.①C语言 – 数据结构 – 程序设计 – 英文②C语言 – 算法分析 – 英文 Ⅳ.TP311.13

中国版本图书馆CIP数据核字（1999）第02420号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：石会敏 版式设计：赵俊斌

北京昌平奔腾印刷厂印刷・新华书店北京发行所发行

2006年2月第1版第5次印刷

787mm × 1092mm 1/16 · 26.5印张

定价：49.00元

To my mother,
Santosh

my wife,
Neeta

and my children,
Agam, Neha, and Param

PREFACE

The study of data structures and algorithms is fundamental to computer science and engineering. A mastery of these areas is essential for us to develop computer programs that utilize computer resources in an effective manner. Consequently, all computer science and engineering curriculums include one or more courses devoted to these subjects. Typically, the first programming course introduces students to basic data structures (such as stacks and queues) and basic algorithms (such as those for sorting and matrix algebra). The second programming course covers more data structures and algorithms. The next one or two courses are usually dedicated to the study of data structures and algorithms.

The explosion of courses in the undergraduate computer science and engineering curriculums has forced many universities and colleges to consolidate material into fewer courses. At the University of Florida, for example, we offer a single one-semester undergraduate data structures and algorithms course. Students coming into this course have had a one semester course in C++ programming and another in discrete mathematics/structures.

Data Structures, Algorithms, and Applications in C++ has been developed for use in programs that cover this material in a unified course as well as in programs that spread out the study of data structures and algorithms over two or more courses. The book is divided into three parts. Part I, which consists of Chapters 1 and 2, is intended as a review of C++ programming concepts and of algorithm analysis and measurement methods. Students who are familiar with programming in C should be able to read Chapter 1 and bridge the gap between

C and C++. Although Chapter 1 is not a primer on C++, it covers most of the C++ constructs with which students might have become rusty. These concepts include modes of parameter passing, template functions, recursion, dynamic memory allocation, classes, and throwing and catching exceptions. More advanced C++ concepts such as inheritence, virtual functions, and abstract classes are described in the chapters where they are first used. Chapter 2 is a review of methods to analyze the performance of a program—operation counts, step counts, asymptotic notation (big oh, omega, theta, and little oh); it also reviews methods to measure performance experimentally. The applications considered in Chapter 2 explore fundamental problems typically studied in a beginning programming course—simple sort methods such as bubble, selection, insertion, and rank (or count) sort; simple search methods such as sequential and binary search; polynomial evaluation using Horner's rule; and matrix operations such as matrix addition, transpose, and multiply. Even though the primary purpose of Chapter 2 is to study performance analysis and measurement methods, this chapter also ensures that all students are familiar with a set of fundamental algorithms.

Chapters 3 through 12 form the second part of the book. These chapters provide an in-depth study of data structures. Chapter 3 forms the backbone of this study by examining various methods of representing data—formula-based, linked, simulated pointer, and indirect addressing. This chapter develops C++ classes to represent the linear list data structure, using each representation method. At the end of the chapter, we compare the different representation schemes with respect to their effectiveness in representing linear lists. The remaining chapters on data structures use the representation methods of Chapter 3 to arrive at representations for other data structures such as arrays and matrices (Chapter 4), stacks (Chapter 5), queues (Chapter 6), dictionaries (Chapters 7 and 11), binary trees (Chapter 8), priority queues (Chapter 9), tournament trees (Chapter 10), and graphs (Chapter 12).

The third part of this book, which comprises Chapters 13 through 17, is a study of common algorithm-design methods. The methods we study are greedy (Chapter 13), divide and conquer (Chapter 14), dynamic programming (Chapter 15), backtracking (Chapter 16), and branch and bound (Chapter 17). Two lower-bound proofs (one for the minmax problem and the other for sorting) are provided in Section 14.4; approximation algorithms for machine scheduling (Section 9.5.2), bin packing (Section 10.5), and the 0/1 knapsack problem (Section 13.3.2) are also covered. NP-hard problems are introduced, informally, in Section 9.5.2.

A unique feature of this book is the emphasis on applications. Several real-world applications illustrate the use of each data structure and algorithm-design method developed in this book. Typically, the last section of each chapter is dedicated to applications of the data structure or design method studied earlier in the chapter. In many cases additional applications are also introduced early in the chapter. We have drawn applications from various areas—sorting (bubble, selection, insertion, rank, heap, merge, quick, bin, radix, and topological sort); matrix algebra (matrix addition, transpose, and multiplication); electronic design automation (finding the nets in a circuit, wire routing, component stack folding, switch-box routing, placement of signal boosters, crossing distribution, and backplane board ordering); compression and coding (LZW compression, Huffman coding, and variable bit-length codes); computational geometry (convex hull and closest pair of points); simulation (machine-shop simulation); image processing (component labeling); recreational mathematics (Towers of Hanoi, tiling a defective chessboard, and rat in a maze); scheduling (LPT schedules); optimization (bin packing, container loading, 0/1 knapsack, and matrix multiplication chains); statistics (histogramming, finding the minimum and maximum, and finding the *k*th smallest); and graph algorithms (spanning trees, components, shortest paths, max clique, bipartite graph covers, and traveling salesperson). Our treatment of these applications does not require prior knowledge of the application areas. The material covered in this book is self-contained and gives students a flavor for what these application areas entail.

By closely tying the applications to the more basic treatment of data structures and algorithm-design methods, we hope to give the student a greater appreciation of the subject. Further enrichment can be obtained by working through the almost 600 exercises in the book and from the associated Web site.

WEB SITE

The URL for the Web site for this book is www.cise.ufl.edu/~sahni/dsac. From this Web site you can obtain all the programs in the book together with sample data and generated output. The sample data is not intended to serve as a good test set for a given program; rather it is just something you can use to run the program and compare the output produced with the given output.

All programs in this book have been compiled and run using Borland's C++ compiler version 5.01 as well as GNU's C++ compiler version 2.7.2.1. The files have been zipped together and placed on the Web site as two separate zip files—one for Borland C++ and the other for GNU C++. The mapping between program numbers in the text and file names is available from the `readme` file, which is included in the zip file.

The Web site also includes solutions to many of the exercises that appear in each chapter, sample tests and solutions to these tests, additional applications, and enhanced discussions of some of the material covered in the text.

ICONS

We have used several icons throughout the book to highlight various features. The icon for a section that provides a bird's-eye view of the chapter contents is

The icon for the explanation of a C++ language construct is

The icon for the treatment of an application is

The icon for a topic on which more material can be found at the Web site is

Some of the exercises have been labeled with the symbol ★. This denotes an exercise whose solution requires development beyond what is done in the chapter. As a result, these exercises are somewhat harder than those without the symbol.

HOW TO USE THIS BOOK

There are several ways in which this book may be used to teach the subject of data structures and/or algorithms. Instructors should make a decision based on the background of their students, the amount of emphasis they want to put on applications, and the number of semesters or quarters devoted to the subject. We give a few of the possible course outlines below. We recommend that the assignments require students to write and debug several programs beginning with a collection of short programs and working up to larger programs as the course progresses. Students should read the text at a pace commensurate with classroom coverage of topics.

TWO-QUARTER SCHEDULE—QUARTER 1

One week of review. Data structures and algorithms sequence.

Week	Topic	Reading
1	Review of C++ and program performance.	Chapters 1 and 2. Assignment 1 given out.
2	Formula-based and linked representations.	Sections 3.1–3.4. Assignment 1 due.
3	Linked, indirect addressing, and simulated pointer.	Sections 3.4–3.7. Assignment 2 given out.
4	Bin sort and equivalence classes.	Section 3.8. Assignment 2 due.
5	Arrays and matrices.	Chapter 4. Examination.
6	Stacks and queues.	Chapters 5 and 6. Assignment 3 given out.
7	Skip lists and hashing.	Chapter 7. Assignment 3 due.
8	Binary and other trees.	Sections 8.1–8.9. Assignment 4 given out.
9	Union/find application. Heaps and heap sort.	Sections 8.10.2, 9.1–9.3, and 9.5.1. Assignment 4 due.
10	Leftist trees, Huffman codes, and tournament trees.	Sections 9.4 and 9.5 and Chapter 10.

TWO-QUARTER SCHEDULE—QUARTER 2

Data structures and algorithms sequence.

Week	Topic	Reading
1	Search trees. Do either AVL or red-black trees. Histogramming.	Chapter 11. Assignment 1 given out.
2	Graphs	Sections 12.1–12.7. Assignment 1 due.
3	Graphs.	Sections 12.8–12.11. Assignment 2 given out.
4	The greedy method.	Sections 13.1–13.3.5. Assignment 2 due.
5	The greedy method and the divide-and-conquer method.	Sections 13.3.6 and 14.1. Assignment 3 given out.
6	Divide-and-conquer applications.	Section 14.2. Examination.
7	Solving recurrences, lower bounds, and dynamic programming.	Sections 14.3, 14.4, and 15.1. Assignment 3 due.
8	Dynamic programming applications.	Sections 15.2.1 and 15.2.2. Assignment 4 given out.
9	Dynamic programming applications.	Sections 15.2.3–15.2.5. Assignment 4 due.
10	Backtracking and branch-and-bound methods.	Chapters 16 and 17.

SEMESTER SCHEDULE
Two weeks of review. Data structures course.

Week	Topic	Reading
1	Review of C++.	Chapter 1. Assignment 1 given out.
2	Review of program performance.	Chapter 2.
3	Formula-based and linked representations.	Sections 3.1–3.4. Assignment 1 due.
4	Linked, indirect addressing, and simulated pointer.	Sections 3.4–3.7. Assignment 2 given out.
5	Bin sort and equivalence classes.	Section 3.8.
6	Arrays and matrices.	Chapter 4. Assignment 2 due. First examination.
7	Stacks. Do one or two applications.	Chapter 5. Assignment 3 given out.
8	Queues. Do two applications.	Chapter 6.
9	Skip lists and hashing.	Chapter 7. Assignment 3 due.
10	Binary and other trees.	Sections 8.1–8.9. Assignment 4 given out.
11	Union/find application.	Section 8.10.2. Second examination.
12	Priority queues, heap sort, and Huffman codes.	Chapter 9. Assignment 4 due.
13	Tournament trees and bin packing.	Chapter 10. Assignment 5 given out.
14	Search trees. Do either AVL or red-black trees. Histogramming.	Chapter 11.
15	Graphs	Sections 12.1–12.7. Assignment 5 due.
16	Graphs. Merge sort and quick sort.	Sections 12.8–12.11, 14.2.2, and 14.2.3.

SEMESTER SCHEDULE

One week of review. Data structures and algorithms course.

Week	Topic	Reading
1	Review of program performance.	Chapters 1 and 2.
2	Formula-based and linked representations.	Sections 3.1–3.4. Assignment 1 given out.
3	Linked, indirect addressing, and simulated pointer.	Sections 3.4–3.8.
4	Arrays and matrices.	Chapter 4. Assignment 1 due.
5	Stacks and queues. Do one or two applications.	Chapters 5 and 6. Assignment 2 given out.
6	Skip lists and hashing.	Chapter 7. First examination. Assignment 2 due.
7	Binary and other trees.	Sections 8.1–8.9. Assignment 3 given out.
8	Union/find application. Heaps and heap sort.	Sections 8.10.2, 9.1–9.3, and 9.5.1.
9	Leftist trees, Huffman codes, and tournament trees.	Sections 9.4 and 9.5 and Chapter 10. Assignment 3 due.
10	Search trees. Do either AVL or red-black trees. Histogramming.	Chapter 11. Assignment 4 given out.
11	Graphs	Sections 12.1–12.7.
12	Graphs and the greedy method.	Sections 12.8–12.11 and 13.1–13.2. Assignment 4 due.
13	Container loading, 0/1 knapsack, shortest paths, and spanning trees.	Section 13.3. Assignment 5 given out.
14	Divide-and-conquer method.	Chapter 14.
15	Dynamic programming.	Chapter 15. Assignment 5 due.
16	Backtracking and branch-and-bound methods.	Chapters 16 and 17.

ACKNOWLEDGMENTS

This book would not have been possible without the assistance, comments, and suggestions of many individuals. I am deeply indebted to the following reviewers for their valuable comments, which have resulted in a better manuscript:

Jacobo Carrasquel	Carnegie Mellon University
Yu Lo Cyrus Chang	University of New Hampshire
Teofilo F. Gonzalez	University of California at Santa Barbara
Laxmikant V. Kale	University of Illinois
Donald H. Kraft	Louisiana State University
Sang W. Lee	University of Michigan
Jorge Lobo	University of Illinois at Chicago
Brian Malloy	Clemson University
Thomas Miller	University of Idaho
Richard Rasala	Northeastern University
Craig E. Wills	Worchester Polytechnic Institute
Neal E. Young	Dartmouth College

Special thanks go to the students in my data structures and algorithms class who provided valuable feedback and helped debug the manuscript. Additionally, I am grateful to the following individuals at the University of Florida for their contributions: Justin Bullard, Edward Y. C. Cheng, Rajesh Dasari, Thomas Davies, Vinayak Goel, Haejae Jung, Jawalant Patel, Sanguthevar Rajasekeran, Gauri Sukhatankar, Gayathri Venkataraman, and Joe Wilson.

The WCB/McGraw-Hill book team has been a pleasure to work with. Everyone contributed immensely to the quality of the final manuscript. The members of this team are Tom Casson, Beth Cigler, Betsy Jones, Brad Kosirog, Madelyn Underwood, John Wannemacher, and Michael Warrell.

Finally, I am indebted to the copy editor, June Waldman, for having done an excellent job.

Sartaj Sahni
Gainesville
October 1997

BRIEF CONTENTS

CONTENTS